Fase 1:

Busqué como usar el circuito integrado “74HC95” y encontré un video de YouTube que podría funcionar parcialmente para la solución, donde usando 3 pines del Arduino 1 generaban un vector fila de matrices en serie, una vez entendiendo el funcionamiento para formar ese arreglo podría extenderse hasta las dimensiones que nos piden en el parcial, por ahora pienso usar 4 integrados “74HC95” para generar la matriz 8x8.

Adjunto evidencia:

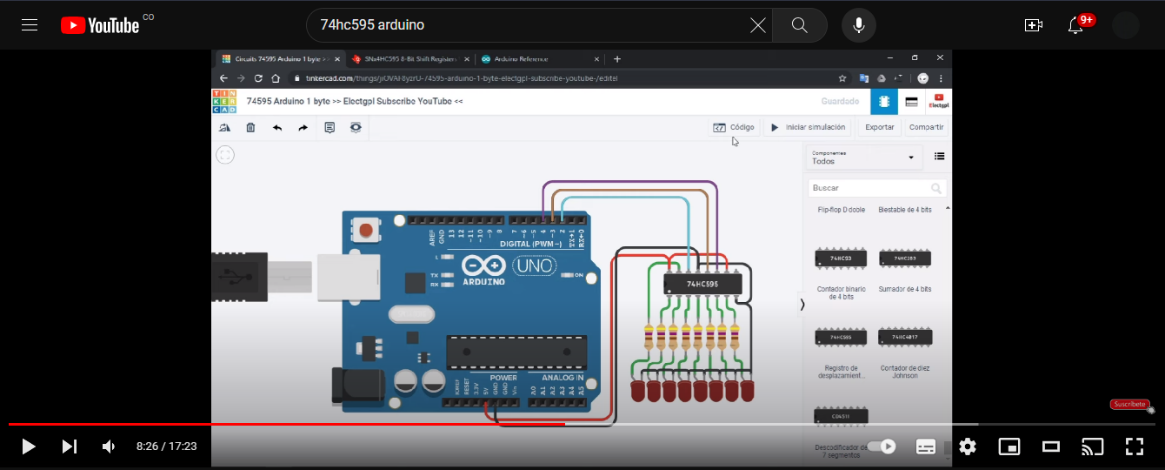


Ilustración 1 Video de YouTube de donde es extrajo la idea  
para más información: [YouTube](https://www.youtube.com/watch?v=ROKculksEr0)

Si bien el primer video me sirvió para empezar aun no me quedaba claro que era el latch y el clock que aparece en el datasheet del circuito integrado, por lo que busque otro video para poder entender mejor el significado, pero ahora también se me genera una duda de como conectar entonces las otras siete filas con 8 leds cada una para poder generar la matriz, por ahora no he empezado, según tengo entendido con el 9 pin, tengo que resolver esta problemática primero para empezar a hacer el código que imprima las figuras, pienso que con un ciclo que retorne 1 y 0 para formar la figura, pero por ahora se queda en un pensamiento.

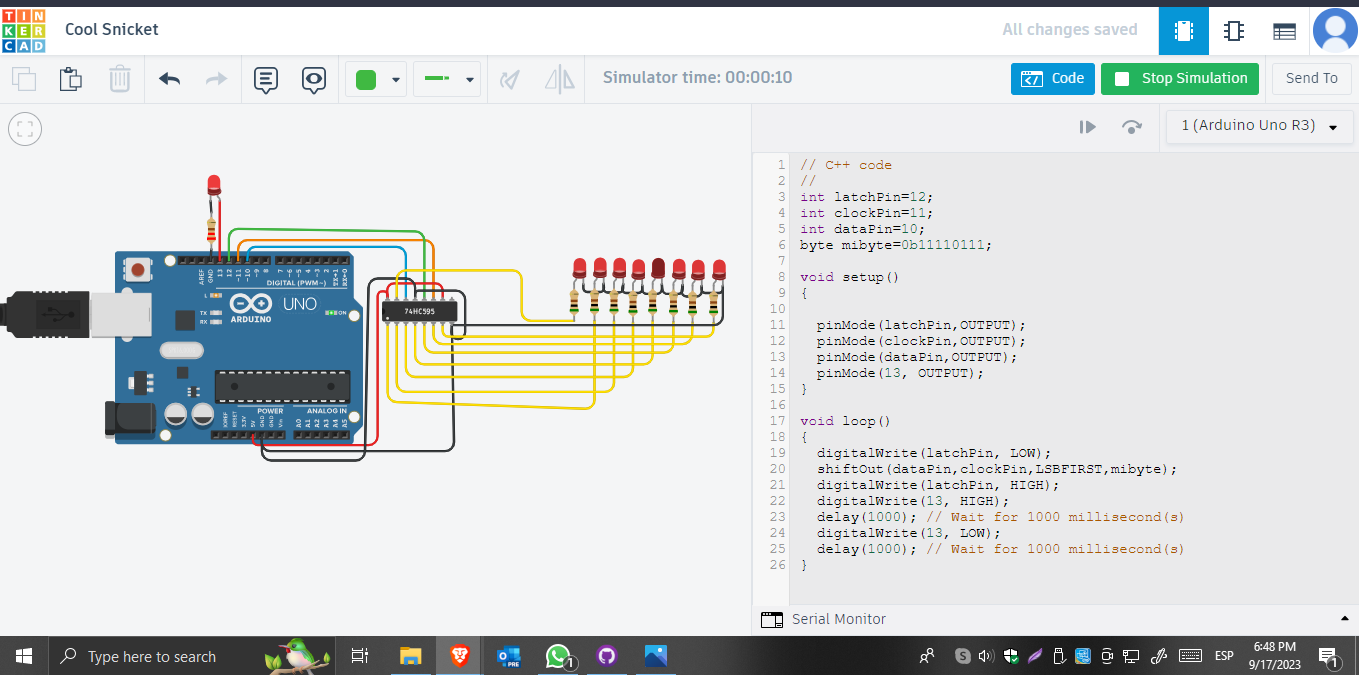


Ilustración 2 Intento primera fila de la matriz, por ahora exitoso.

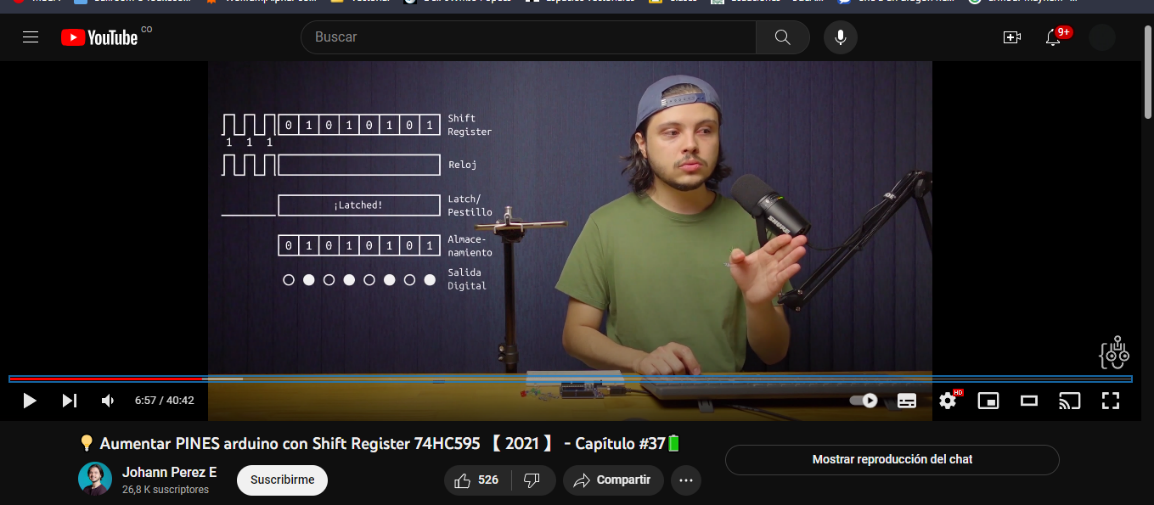


Ilustración 3 Segundo video de referencia, para más información: [YouTube](https://www.youtube.com/watch?v=-pvTlcFQ2Wg).

Pude armar la matriz colocando 8 integrados, por un momento pensé que lo podía hacer con 4 porque confundí el tamaño de la matriz quizás podría usar los otros 3 pines disponibles del Arduino pero por ahora con los 8 integrados me es suficiente, también se como acceder a ellos y al menos establecer la solución trivial para los patrones, como ya culminé el armado de la matriz ahora puedo enfocarme en el código, estaba pensando en crear un arreglo tipo byte de tamaño 8 en el cual ir almacenado los 1s y 0s generados por las funciones de los patrones y una vez guardados en un byte cada fila pasarlas por un ciclo for que las vaya mostrando en el montaje.

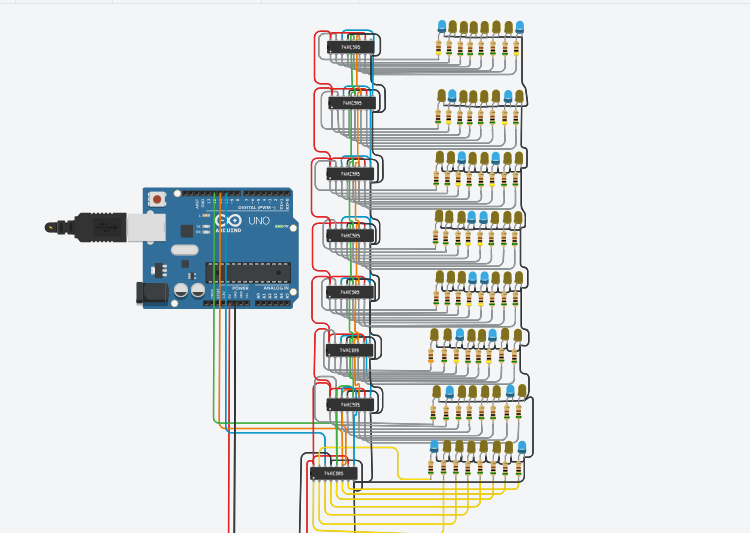


Ilustración 4 Montaje de leds 8x8 encendido con patrón cruz mediante solución trivial, el cambio de color de led es para diferenciar los encendidos(azules) de los pagado(amarillos)s.

Se usó este código:

// C++ code

//

int latchPin=12;

int clockPin=11;

int dataPin=10;

byte mibyte=0b10000001;

byte mibyte2=0b01000010;

byte mibyte3=0b00100100;

byte mibyte4=0b00011000;

byte mibyte5=0b00011000;

byte mibyte6=0b00100100;

byte mibyte7=0b01000010;

byte mibyte8=0b10000001;

void setup()

{

pinMode(latchPin,OUTPUT);

pinMode(clockPin,OUTPUT);

pinMode(dataPin,OUTPUT);

pinMode(13, OUTPUT);

}

void loop()

{

digitalWrite(latchPin, LOW);

shiftOut(dataPin,clockPin,LSBFIRST,mibyte);

shiftOut(dataPin,clockPin,LSBFIRST,mibyte2);

shiftOut(dataPin,clockPin,LSBFIRST,mibyte3);

shiftOut(dataPin,clockPin,LSBFIRST,mibyte4);

shiftOut(dataPin,clockPin,LSBFIRST,mibyte5);

shiftOut(dataPin,clockPin,LSBFIRST,mibyte6);

shiftOut(dataPin,clockPin,LSBFIRST,mibyte7);

shiftOut(dataPin,clockPin,LSBFIRST,mibyte8);

digitalWrite(latchPin, HIGH);

}

Fase 2:

* 1. Hoy me concentraré en hacer los códigos que permitan dibujar los patrones, por ahora mi idea es crear un arreglo donde se guarden la información de las filas( en este caso donde se guardan los números 1) con esa información tendría en que posición se encuentran los unos y asi por conversión podría decir que leds encender o no, también hay una cuestión de orden al final usare, si con el MSB o LSB, pero esto ultimo explicado seria la segunda fase de la segunda fase de la implementación del código.
  2. Se diseñaron al menos dos patrones, por ahora es suficiente, con ellos ya podría empezar a crear el código para almacenar la información y finalmente generar la imagen en los leds, aquí adjunto el código para los primeros dos patrones:

1. #include<iostream>
2. using namespace std;
4. int main()
5. {
6. //Primer patron:
7. for(int filas=0;filas<4;filas++){
9. for(int columnas=0;columnas<4;columnas++){
10. if(columnas<3-filas){
11. cout<<'-';
12. }
13. else cout<<'\*';
14. }
15. for(int columnas=0;columnas<4;columnas++){
16. if(columnas<=filas){
17. cout<<'\*';
18. }
19. else cout<<'-';
20. }
21. cout<<endl;
23. }
24. for(int filas=0;filas<4;filas++){
26. for(int columnas=0;columnas<4;columnas++){
27. if(columnas>=filas){
28. cout<<'\*';
29. }
30. else cout<<'-';
31. }
32. for(int columnas=0;columnas<4;columnas++){
33. if(columnas<4-filas){
34. cout<<'\*';
35. }
36. else cout<<'-';
37. }
38. cout<<endl;
40. }
41. //Patron2
42. for(int filas=0;filas<4;filas++){
44. for(int columnas=0;columnas<4;columnas++){
45. if(columnas==filas){
46. cout<<'\*';
47. }
48. else cout<<'-';
49. }
50. for(int columnas=0;columnas<4;columnas++){
51. if(columnas==3-filas){
52. cout<<'\*';
53. }
54. else cout<<'-';
55. }
56. cout<<endl;
58. }
59. for(int filas=0;filas<4;filas++){
61. for(int columnas=0;columnas<4;columnas++){
62. if(columnas==3-filas){
63. cout<<'\*';
64. }
65. else cout<<'-';
66. }
67. for(int columnas=0;columnas<4;columnas++){
68. if(columnas==filas){
69. cout<<'\*';
70. }
71. else cout<<'-';
72. }
73. cout<<endl;
75. }

78. return 0;
79. }

Mi idea para almacenar los datos es crear un arreglo de 64 posiciones y moverme dentro de el en intervalos de 8 para asi moverme entre filas, también creando un arreglo temporal donde voy a guardar la información de la fila que estoy examinando para asi generar el número que en principio estaría en binario a su representación decimal para así pasarlo a los leds.

Tuve que actualizar el código para Finalmente guardar la representación de los números binarios en decimal exitosamente, por medio de guardar todos los patrones en un arreglo de 64 caracteres y contando los asteriscos que lo forman que serían el equivalente de los 1 en binarios, con esto encendiendo los leds necesarios para cada fila. El código es el siguiente:

    char arreglo[64]={0};

    int contador=0,contador\_2=0,contador\_3=0,numerobinario=0, arreglo\_2[8]={0};

    //Primer patron:

    for(int filas=0;filas<4;filas++){

        for(int columnas=0;columnas<4;columnas++){

            if(columnas<3-filas){

                arreglo[contador]='-';

                cout<<'-';

            }

            else{

                arreglo[contador]='\*';

                cout<<'\*';

            }

            contador++;

        }

        for(int columnas=0;columnas<4;columnas++){

            if(columnas<=filas){

                arreglo[contador]='\*';

                cout<<'\*';

            }

            else{

                cout<<'-';

                arreglo[contador]='-';

            }

            contador++;

        }

        cout<<endl;

    }

    for(int filas=0;filas<4;filas++){

        for(int columnas=0;columnas<4;columnas++){

            if(columnas>=filas){

                arreglo[contador]='\*';

                cout<<'\*';

            }

            else {

                arreglo[contador]='-';

                cout<<'-';

            }

            contador++;

        }

        for(int columnas=0;columnas<4;columnas++){

            if(columnas<4-filas){

                arreglo[contador]='\*';

                cout<<'\*';

            }

            else {

                arreglo[contador]='-';

                cout<<'-';

            }

            contador++;

        }

        cout<<endl;

    }

while(contador\_2<contador){

    numerobinario=0;

    for(int i=0;i<8;i++){

        if(arreglo[contador\_2]=='\*'){

            numerobinario=numerobinario+pow(2,i);

        }

        contador\_2++;

    }

    arreglo\_2[contador\_3]=numerobinario;

    contador\_3++;

}

cout<<contador;

Esta es una versión modificada del anterior donde tuve que incluir la librería math.h para guardar los equivalentes de binarios en enteros usando la funcion pow(), en este código me limite a usar únicamente el primer patron para facilitar la ejecución, con esto claro ahora explico las nuevas variables y su cometido en el código:

**arreglo[64]**  es un arreglo de tipo char donde se van a almacenar las filas de los patrones linealmente, por lo que al final estaría almacenado 64 caracteres.

**Contador:**  es para usarlo como índice de acceso a arreglo de esta forma arreglo[contador] y asi de esta forma el carácter que se generó en la ejecución puede ser almacenado en arreglo y una vez lleno poder acceder a ellos nuevamente. También se tiene que tener en cuenta que este contador debe ir aumentando de 1 en 1 hasta 64 para asi asegurar que ha pasado por todos los caracteres o en otras palabras por todos los leds.

**contador2:** se usa como condición de mientras para el while y va desde 1 hasta 64.

**contador\_3:** su funcion es exactamente la misma que la de contador, solo que esta vez es para guardar el equivalente de binario en decimal en arreglo2.

**numero\_binario:**esta es simplemente para ir almacenado el equivalente de binario que se tiene hasta que se completa la primera fila para luego hacer arreglo2[contador\_3]=numero\_binario.Este se debe inicializar en 0 una vez se termina el primer ciclo for.

Por ahora usar memoria dinámica no lo veo viable porque realmente la interacción con los inputs del usuario no lo requieren, pero si podría usar funciones y/o punteros para una programación mas limpia y ordenada.